LA-UR-91-607

LA-UR--91-607

DE91 008594

TITLE A Parallel Non-neural Trigger Tracker for the SSC

AUTHOR(S) R.m. Farber, W. Kennison, A.S. Lapedes
Los Alamos National Laboratory

SUBMITTED TO IJCNN-91-Seattle Conference
July 8-11, 1991

A Parallel Non-Neural Trigger Tracker for the SSC

R.M. Farber, W Kennison, and A.S. Lapedes
Los Alamos National Laboratory
M.S. B213
Los Alamos, N.M. 87545

ABSTRACT

The Superconducting Super Collider (SSC) is a major project promising to open the vistas of very high energy particle physics. When the SSC is in operation, data will be produced at a staggering rate. Current estimates place the raw data coming out of the proposed silicon detector system at 2.5×10^{16} bits/second. Clearly, storing all events for later off-line processing is totally impracticable. A hierarchy of triggers, firing only on events meeting increasingly specific criteria, are planned to cull interesting events from the flood of information. Each event consists of a sequence of isolated "hits", caused by particles hitting various parts of the detector. Collating these hits into the tracks of the approximately 500 particles/event, and then quickly deciding which events meet the criteria for later processing, is essential if the SSC is to produce usable information. This paper addresses the need for real-time triggering and track reconstruction. A benchmarked and buildable algorithm, operable at the required data rates, is described. The use of neural nets, suggested by other researchers, is specifically avoided as unnecessary and impractical. Instead, a parallel algorithm, and associated hardware architecture using only conventional technology, is presented. The algorithm has been tested on fully scaled up, extensively detailed, simulated SSC events, with extremely encouraging results. Preliminary hardware analysis indicate that the trigger/tracker may be built within proposed SSC budget guidelines.

INTRODUCTION

It has been proposed that neural networks, due to their massively parallel analogue nature, are well suited to quickly reconstructing the particle tracks generated during an SSC event [1,2,3]. These proposals have only been tested on toy data, and little thought has been given to the extreme difficulties of building a neural network hardware implementation capable of operating at the required data rates. We take the view that neural networks are not necessary, if not impractical, for reconstructing an SSC event. We propose a simple algorithm/parallel systems architecture which reconstructs the particle tracks given event data from the silicon detector system, and which is capable of triggering on interesting events using operations no more complex than integer comparison and table lookup. We have benchmarked this algorithm on realistic, simulated data of the expected size and detail of real SSC events. We show that it is possible to build a parallel hardware implementation of the algorithm which would be fast enough to examine the raw data in real-time, and trigger on events with interesting phenomena, such as Higgs events. We also show the algorithm is capable of reconstructing with high accuracy particle tracks over a specified $P_\perp$ threshold. ($P_\perp$ is a measure of the particle momentum transverse to the magnetic field. High $P_\perp$ events are of interest.) Simplicity in computation and communication within the proposed hardware implementation indicate that the system may be built at an acceptable cost.

DESCRIPTION OF THE SSC

The SSC will be a proton-on-proton colliding beam machine. The machine will produce interactions with a center of mass energy of 40 TeV and with a luminosity of 10^{33}/cm^2/sec. Beam crossings will occur every 16 ns. For these studies, we have used as our detector model one of the designs of the silicon barrel for the L* detector [4]. That detector model has six evenly spaced concentric silicon shells. The innermost shell has a radius of 10 cm and the outermost shell has a radius of 35 cm. Each detector shell is made of silicon wafers with nominal dimensions of 3.0-cm width, 6-cm length and 280-μm thickness. The silicon wafers have double sided readout on 50 μm pitch for the r-ϕ measurements and 5-mrad stereo angle to determine the z (along the beam direction) position. The readout of the silicon is every 18 cm along the z direction. The detector simulation includes a fair representation of the silicon response function for charged particles, including a 0.5-mm dead region around every edge of the silicon wafers and a 20 KeV energy deposition threshold per strip. For each hit the data from the detector will consist of the address of the first strip in a cluster of struck strips and the width of the cluster. The coordinates of the hit, which inputs to the trigger and track reconstruction code, are then determined from the cluster information.

THE TRACK RECONSTRUCTION ALGORITHM

The algorithm is presented with a number of addresses from each shell of detectors. Each address represents a detector which has been triggered by either a particle passing through the detector, or by noise. The location of each detector in ϕ and Z is known from it's 14 bit address. Given the size and location of the "reaction region", i.e. the region from which particles may emanate, and a minimum threshold $P_\perp$, it is possible to define a region, or road, which will contain any particle track greater than or equal to the minimum $P_\perp$. Due to ambiguity in specifying a set size from which the majority of particles emanate, the boundaries of the road were determined empirically. Centering a road on each hit in the outer shell, and the known reaction region, defines the possible detector address in each shell which could be triggered by a track above threshold (See Fig. 1).

For each point on the outermost shell, 5, the boundaries of the road define all the points next innermost shell, 4, which must be examined to find any tracks greater than or equal to the threshold $P_\perp$. It is possible to drastically reduce the search region for the next inner shell, 3, by assuming (incorrectly) that the track originated from the center of the reaction region and passed through the points found in shells 4 and 5. This need not actually happen, but it is useful to define this idealized track as a computational artifice. This fictious path is uniquely determined by the three points ("hit on origin", "hit" on shell 4, "hit" on shell 5). The position of the detector on shell three nearest the "hit" from this "track" may be easily found. A range of shell 3 detector addresses can now be defined by considering a tube of some specified radius surrounding the fictitious "track", and the resulting range of addresses on shell three. The radius of the tube is empirically determined to account for particles which, in fact, emanated from the extremes of the reaction region. Once a point in shell 3 is known, it is possible to estimate very well, without assumptions, the unique path the particle had to take from the reaction region through all the shells. It is then a simple matter to search shells 0,1 and 2 for address which fit within the projected path of the particle. The projected path is, however, only an estimation since the points used to specify the path are only approximately known. This is due to the finite number of detectors per shell which induces an uncertainty in the actual position of each hit. To handle this issue of granularity of the detectors, the search region is artificially expanded by a specified amount which is related to the detector granularity. The size of this expanded search region is again determined empirically from simulated SSC events and the known detector geometry. Only those detector addresses which in combination through all the shells define a physically possible particle path , with $P_\perp$ greater than a threshold, are called a track. The rest are discarded as either noise or particle tracks below the threshold $P_\perp$. All empirically determined quantities have been chosen to produce results we deemed as acceptable and none have been set "optimally".

Instead of building the entire track reconstruction algorithm in hardware to reconstruct every track in an event, we propose a simpler version of the algorithm that triggers only on "interesting" tracks. Only these "interesting" events are passed to the track reconstruction algorithm, allowing it to be slower and less expensive. The details of the possible hardware implementation of the full track reconstruction algorithm are not presented in this paper due to space restrictions.

The Trigger Algorithm

The algorithm used in triggering is a very simple version of the track reconstruction algorithm. As can be seen from Figs. 2-4, the most complex computations within the algorithm are table lookup and integer comparison. Additionally, the hardware units are linked via the simplest forms of interprocessor communications: global broadcast of values from one unit to many and local communications from one unit to another.

Initially the trigger algorithm starts out the same as the track reconstruction algorithm by defining a roadway centered on each outer shell detector hit and running between the outer shell hit and the reaction region. For triggering, a parallel search is then made to see if there are any detector hits on the next two inner shells that are within the roadway boundaries. Any outer shell hits which have corresponding hits within the boundaries of the roadway on the next two inner shells are counted as tracks. A count is then made to see if enough tracks have been found to cause a trigger. In the case of a Higgs event, two or more tracks must be found. It should be clear that too low a choice for the threshold $P_\perp$ will cause excessive triggering. We have found that even a low threshold $P_\perp$, around 5 Gev, works quite well. The selection of the correct threshold $P_\perp$, as we pointed out earlier, is subject to optimization.

THE TRIGGER HARDWARE

As can be seen in Fig. 2 each address from a detector hit on the outermost shell goes to a separate computational block. Simulation results show that 500 computational blocks will be required. All addresses from detector hits in the next two inner shells, 3 and 4, are also loaded in parallel into all computational blocks. A boolean "Found Track" line comes out of each computational block and into a high speed comparison circuit which determines if enough tracks have been found to cause triggering. This comparison circuit could be as simple as a multi-line OR gate or as sophisticated as a high speed counter/comparator. In either case, a trigger will occur if enough computational blocks signal they have a track greater than or equal to the threshold $P_\perp$.

Figure 3 shows the detail of the computational block. The addresses of all the hits in each shell are broadcast in parallel into an array of search blocks - one array for each shell and one search block for each address.. The address of the outermost detector is used in each computational block to index into a lookup table to find the appropriate search boundaries for each shell. Each table lookup occurs simultaneously and the resulting boundary is globally broadcast into the appropriate array of search blocks. If a search block finds a detector address within the bounds broadcast from the table lookup, it sets a boolean line high to indicate a match. If one or more search blocks in both shells indicate a match and an AND gate can be used to indicate that a track has been found..

It is worth pointing out that the lookup tables are quite small, containing less than 50,000 memory locations storing 14 bits. Since the addresses are consecutive, current 16 ns RAM chips would be an obvious choice for the hardware implementation. The lookup tables are initialized once at start-up and can be used to account for inter-shell as well as intra-shell detector misalignments.

Details of the search block are shown in fig. 4. For simplicity, only a one dimensional search block has been diagrammed. The hardware to search in two dimensions, including wraparound of the ϕ coordinates, is easily constructed (without adding additional computational cycles) by combining multiple one dimensional search blocks together.

The operation of the search block is as follows: In one clock cycle, two registers are loaded with the address from a single detector in the appropriate shell. In the next clock cycle, the upper and lower search bounds are each loaded into separate registers. Two simultaneous comparisons occur in the next clock cycle to see if the detector address is between the upper and lower bounds. If both comparisons are found to be true then both lines from the comparators will be true and the AND gate will signal a match.

Fig. 4 shows that with counting two cycles for table lookup, determining a track greater than or equal to the threshold $P_{\perp}$ will take 4 cycles. Determining if there are enough found tracks to cause a trigger should take one cycle and dumping the event to a buffer for either storage or full scale reconstruction should also take one cycle (remember the data was loaded in one cycle). The total number of cycles to load, examine, trigger, and dump an event will take 6 cycles. The slowest cycle time required to load the data is 16ns, which is easily met by current technology. This means that we have a worst cast runtime of 6*16 ns or 96 ns. A real implementation will probably run faster due to a number of optimizations not presented in this paper to preserve clarity.

To process all the data coming out of the SSC in real time, it is necessary to buffer the data.. A number of these triggers can be arranged in a circular buffer. Then, one trigger after another is loaded and processing initiated . Since the number of cycles required to process and dump the data is known, enough triggers can placed in the circular buffer to ensure that each trigger has finished before it is presented with a new SSC event. Best estimates for the necessary hardware indicate the system can be built within SSC budget guidelines.

SIMULATIONS AND RESULTS

The input for the trigger and track reconstruction simulator are the output coordinates from the detector simulator, as previously described. The event simulator is based upon GEANT 3 [5] and uses PYTHIA [6] for event generation. Two different data samples are simulated. In the first, PYTHIA minimum bias data are generated. Those data represent the events which comprise the main physics background at the SSC. The trigger should reject all of the minimum bias events. For an example of interesting events, we have generated Higgs events which the trigger should accept with high efficiency. The data used in the simulations reported here consisted of 300 GeV/c^2 Higgs decaying into two Z^0's which in turn decay to $\mu\mu$- and ee-pairs, respectively. (These are known as so called "gold-plated Higgs" events.) The GEANT 3 Monte Carlo procedure tracks all charged particles through the detector and known physical processes associated with the passage of charged particles through matter are simulated. The resulting coordinates, therefore, are a fair approximation of what will be seen in a true SSC detector.

The algorithms were tested with the above simulated data with approximately

500 tracks per event, complete with complications caused by multiple scattering, false detector hits, and low momentum particles, to name but a few. Current accuracies on simulated "real world" SSC events indicates that the algorithm triggers correctly on Higgs events 98.47% of the time, while correctly discarding uninteresting minimum bias events 99.9% of the time. In reconstructing any given SSC event, either Higgs or minimum bias, the algorithm reproduces 97.8% of the particle tracks over 500 Mev $P_{\perp}$, with 0.62 spurious tracks falsely generated per each track that should be found. We have considered the selection of Higgs events as an example of interest to many people, but this should not be considered the only application of the system.

CONCLUSIONS

It is expected that the behavior of the system could be much more finely tuned once the scientific requirements of the SSC are more clearly defined, although the current implementation seems to satisfy the stringent requirements for a trigger and tracker. The choice of the P threshold is critical to the application of the system. Higher minimum threshold $P_{\perp}$ values have the desirable effect of cutting down on the number of spurious tracks found, while preserving a high accuracy of track reconstruction. Setting a high $P_{\perp}$ would allow triggering only on events with high momentum tracks, or on events which have greater than a desired count of high momentum tracks. The results we have to date use a reasonable P threshold and are very encouraging, although the current statistics are based on a relatively small number of examples, namely 1500 Higgs events and 5512 minimum bias events. Comparisons to other systems are not possible at this stage, as we know of no other algorithm that has been tested on "real" data (i.e. extremely realistic, simulated SSC events). Fewer still are algorithms where attention has been paid to practical issues of buildability. A trigger system could be built using current commercial technology for less than the cost of a modern supercomputer [7]. This system can provide a benchmark for other methods, which when developed, can realistically handle data of the complexity and volume of "real time" "real world" SSC events.

ACKNOWLEDGMENTS

We are grateful for the help of Pat McGaughey in verifying the hardware figures. We thank David Wolf for providing code which fits and predicts the particle paths. We also would like to thank the Santa Fe Institute where part of this work was performed. This work was performed under the auspices of the U.S. Department of Energy.

REFERENCES

1) "Neural Networks for Triggering, B. Denby *et. al.*, Preprint Fermi National Accelrator Laboratory, 1990.

2) "Tests of Track Segment and Vertex Finding With Neural Networks",B. Denby *et. al.* presented at the "Conference on COmputing in High Energy Physics", Santa Fe, N.M., April 1990.

3) "Data Processing at the SSC with Structered Neural Nets" Lackner K.S. *et.al.*, LA-UR-90-3774, Oct. 1990.

4) "Expression of Interest to the Superconducting Super Collider Laboratory", L* Collaboration, private communication.

5) "GEANT 3", R.Brun, *et.al.*, CERN Report DD/EE/84-1 (Sept. 1987)

6) "The LUND Monte Carlo for Hardware Processes-PYTHEA Version 5.3", Hans-Uno Bergtsson and Torbjörn Sjöstrand, DESY Report LU TP 87-3, (Nov.,1989).

7) "XILINX Technical Data XC4000 Logic Cell Array Family", 2100 Logic Drice, San Jose, Calif. 95124, 1990.

Fig 1

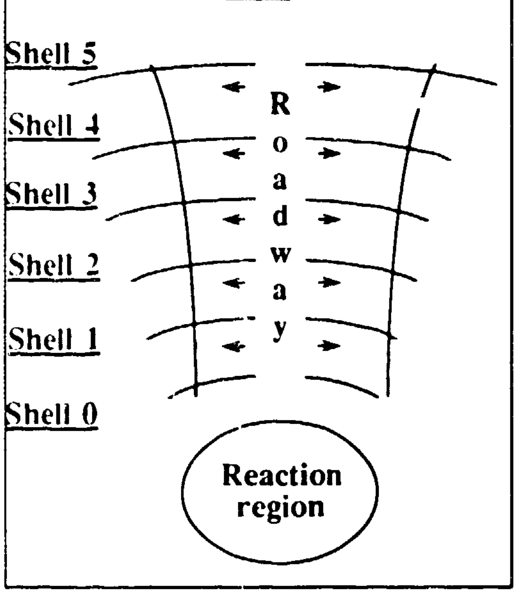

Fig 2 TRIGGER: FUNCTIONAL OUTLINE

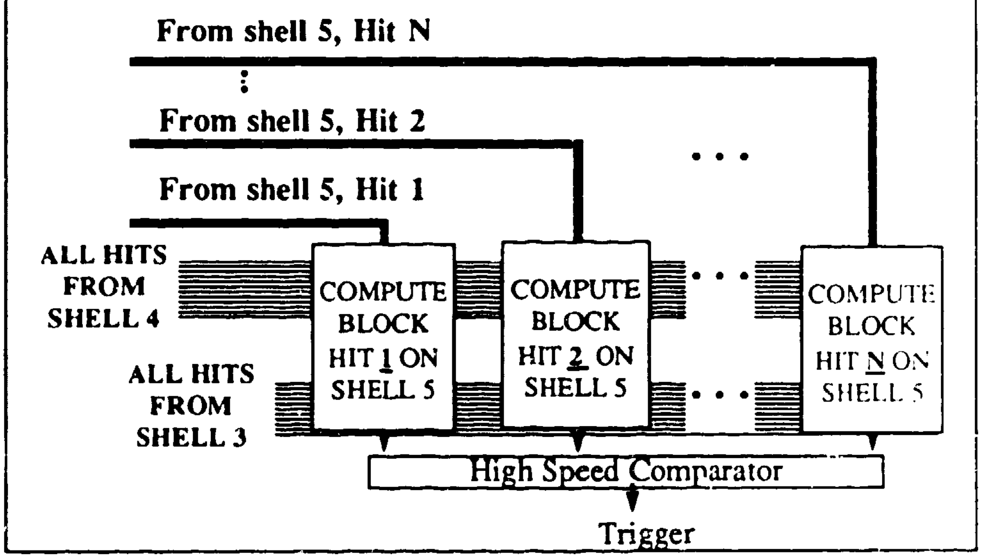

Fig 3 COMPUTE BLOCK: FUNCTIONAL OUTLINE

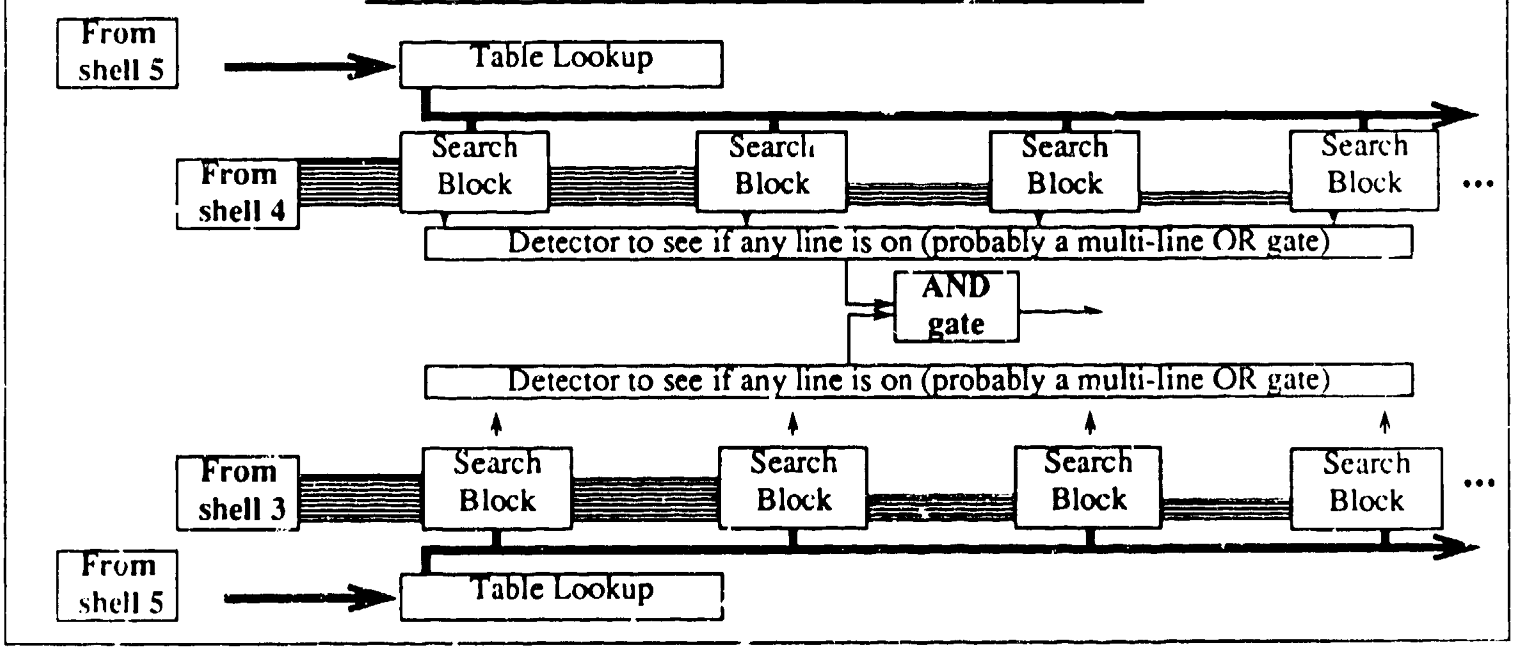

Fig 4 SEARCH BLOCK: FUNCTIONAL OUTLINE